

862.C1901



PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

MIYUKI ENOKIDA, ET AL.

Application No.: 09/558,656

Filed: April 26, 2000

For: DATA PROCESSING METHOD  
AND DATA PROCESSING  
DEVICE

Examiner: N/Y/A

Group Art Unit: 2771

Date: September 8, 2000

TC 2700 MAIL ROOM

SEP 15 2000

RECEIVED

2771  
K. W. Lind  
9/20/00  
#3  
Printy  
paper

Commissioner for Patents  
Washington, D.C. 20231

CLAIM TO PRIORITY

Sir:

Applicants hereby claim priority under the  
International Convention and all rights to which they are  
entitled under 35 U.S.C. § 119 based upon the following  
Japanese Priority Applications:

2000-109923 filed on April 11, 2000; and

11-120713 filed on April 27, 1999.

Certified copies of the priority documents are  
enclosed.

Applicants' undersigned attorney may be reached in our New York office by telephone at (212) 218-2100. All correspondence should continue to be directed to our address given below.

Respectfully submitted,

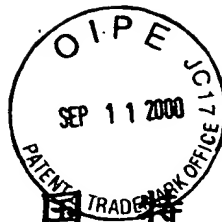


Attorney for Applicants

Registration No. 28 286

FITZPATRICK, CELLA, HARPER & SCINTO  
30 Rockefeller Plaza  
New York, New York 10112-3801  
Facsimile: (212) 218-2200

NY\_MAIN 81684 v2



CFM 1901 US  
USAN 09/558,656

日 本 国 特 許 庁

PATENT OFFICE  
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日  
Date of Application:

2000年 4月11日

出 願 番 号  
Application Number:

特願2000-109923

出 願 人  
Applicant(s):

キヤノン株式会社

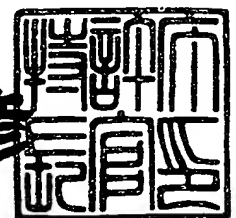
RECEIVED  
SEP 15 2000  
2700 MAIL ROOM

CERTIFIED COPY OF  
PRIORITY DOCUMENT

2000年 5月19日

特許庁長官  
Commissioner,  
Patent Office

近 藤 隆 彦



出証番号 出証特2000-3036129

【書類名】 特許願

【整理番号】 4190007

【提出日】 平成12年 4月11日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 3/00

【発明の名称】 データ処理方法及び装置及び記憶媒体

【請求項の数】 61

【発明者】

    【住所又は居所】 東京都大田区下丸子3丁目30番2号 キヤノン株式会社  
社内

    【氏名】 榎田 幸

【発明者】

    【住所又は居所】 東京都大田区下丸子3丁目30番2号 キヤノン株式会社  
社内

    【氏名】 草間 澄

【特許出願人】

    【識別番号】 000001007

    【氏名又は名称】 キヤノン株式会社

【代理人】

    【識別番号】 100076428

    【弁理士】

    【氏名又は名称】 大塚 康德

    【電話番号】 03-5276-3241

【選任した代理人】

    【識別番号】 100101306

    【弁理士】

    【氏名又は名称】 丸山 幸雄

    【電話番号】 03-5276-3241

【選任した代理人】

【識別番号】 100115071

【弁理士】

【氏名又は名称】 大塚 康弘

【電話番号】 03-5276-3241

【先の出願に基づく優先権主張】

【出願番号】 平成11年特許願第120713号

【出願日】 平成11年 4月27日

【手数料の表示】

【予納台帳番号】 003458

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0001010

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 データ処理方法及び装置及び記憶媒体

【特許請求の範囲】

【請求項 1】 ディレクトリ構造でファイルを管理するファイル管理システムにおいてディレクトリに対応するディレクトリデータを読み込む第 1 読込工程と、

前記ディレクトリデータに付与すべきメタデータを読み込む第 2 読込工程と、

前記第 1 読込工程で読み込まれたディレクトリデータの後に、前記第 2 読込工程で読み込まれたメタデータを接続する接続工程と、

前記接続工程によって得られたデータの全体をディレクトリデータファイルとして出力する出力工程と

を備えることを特徴とするデータ処理方法。

【請求項 2】 前記第 2 読込工程で読み込まれたメタデータが、所定のデータ記述言語における適正な形式で記述されているか否かを判定する判定工程を更に備え、

前記接続工程は、前記判定工程で適正な形式で記述されていると判定された場合に、前記メタデータを前記ディレクトリデータの後に接続する

ことを特徴とする請求項 1 に記載のデータ処理方法。

【請求項 3】 前記判定工程は、前記メタデータが前記所定のデータ記述言語としての正当性を満足するか否かに基づいて判定する

ことを特徴とする請求項 2 に記載のデータ処理方法。

【請求項 4】 前記判定工程は、前記メタデータがデータ記述言語が文法的に正しいか否かに基づいて判定する

ことを特徴とする請求項 2 に記載のデータ処理方法。

【請求項 5】 ディレクトリデータファイルのデータにメタデータが登録されているか否かを判別するデータ処理方法であって、

ディレクトリデータファイルを読み込む読込工程と、

前記読込工程で読み込まれたディレクトリデータファイルのデータを末尾より検査し、所定のデータ記述言語における適正な形式で記述されたデータが存在す

るか否かを判定することにより、該データに含まれるメタデータを判別する判別工程と

を備えることを特徴とするデータ処理方法。

【請求項 6】 前記判別工程においてメタデータが判別された場合、判別されたメタデータを抽出して出力する出力工程を更に備える

ことを特徴とする請求項 5 に記載のデータ処理方法。

【請求項 7】 前記出力工程は、前記抽出されたメタデータに基づく表示を行う

ことを特徴とする請求項 6 に記載のデータ処理方法。

【請求項 8】 前記出力工程は、前記抽出されたメタデータを、前記所定のデータ記述言語に基づいて所定の処理を実行するためのツールに提供する

ことを特徴とする請求項 6 に記載のデータ処理方法。

【請求項 9】 前記判別工程は、

前記所定のデータ記述言語によって規定された末尾文字列が前記データの末尾に存在するか否かをチェックするチェック工程と、

該末尾文字列が存在する場合に前記所定のデータ記述言語に規定された先頭文字列を該データの先頭へ向かって検索する検索工程とを備え、

前記末尾文字列と前記先頭文字列によって挟まれたデータが存在する場合に、このデータをメタデータと判別する

ことを特徴とする請求項 5 に記載のデータ処理方法。

【請求項 10】 前記判別工程は、前記末尾文字列と前記先頭文字列によって挟まれたデータが、前記所定のデータ記述言語における適正な形式を有するか否かを検査する検査工程を更に備える

ことを特徴とする請求項 9 に記載のデータ処理方法。

【請求項 11】 前記検査工程は、前記所定のデータ記述言語としての正当性を満足するかの判断を含めて検査する

ことを特徴とする請求項 10 に記載のデータ処理方法。

【請求項 12】 前記検査工程は、前記所定のデータ記述言語が文法的に正しいか否かの判断を含めて検査する

ことを特徴とする請求項10に記載のデータ処理方法。

【請求項13】 処理対象データが属するディレクトリに対応するディレクトリデータを読み込む読込工程と、

前記読込工程で読み込まれたディレクトリデータよりメタデータを抽出する抽出工程と、

前記抽出工程で抽出されたメタデータに基づいて、前記処理対象データへメタデータを付加する付加工程と

を備えることを特徴とするデータ処理方法。

【請求項14】 前記付加工程は、

前記処理対象データにメタデータが登録されているか否かを判定する判定工程を備え、

メタデータが登録されていないと判定された場合に、前記処理対象データに前記抽出工程で抽出されたメタデータを付加する

ことを特徴とする請求項13に記載のデータ処理方法。

【請求項15】 前記付加工程は、

前記処理対象データにメタデータが登録されているか否かを判定する判定工程と、

メタデータが登録されていると判定された場合に、前記処理対象データからメタデータを分離する分離工程と、

前記分離工程で得られたメタデータと、前記抽出工程で得られたメタデータとに基づいて新たなメタデータを生成する生成工程とを備え、

前記生成工程によって生成されたメタデータを、前記分離工程によってメタデータが分離された後の前記処理対象データに付加する

ことを特徴とする請求項13に記載のデータ処理方法。

【請求項16】 前記生成工程は、前記分離工程で得られたメタデータと、前記抽出工程で得られたメタデータとに含まれる全てのデータ項目が含まれるように新たなメタデータを生成する

ことを特徴とする請求項15に記載のデータ処理方法。

【請求項17】 前記処理対象データは、他のディレクトリにコピーされた



データである

ことを特徴とする請求項13に記載のデータ処理方法。

【請求項18】 前記処理対象データは、他のディレクトリに移動されたデータである

ことを特徴とする請求項13に記載のデータ処理方法。

【請求項19】 前記処理対象データはバイナリデータ部分を含み、  
前記付加工程は、前記処理対象データのバイナリデータ部分の後にメタデータを接続する

ことを特徴とする請求項13に記載のデータ処理方法。

【請求項20】 前記処理対象データは、画像データ、音声データ、動画データのうちいずれかである

ことを特徴とする請求項13に記載のデータ処理方法。

【請求項21】 指定されたディレクトリ内に属するデータファイルを読み込む読込工程と、

前記読込工程で読み込まれたデータファイルからメタデータを抽出する抽出工程と、

前記抽出工程で抽出されたメタデータに基づいてディレクトリ用のメタデータを生成する生成工程と、

前記生成工程で生成されたメタデータをディレクトリデータに付加する付加工程と

を備えることを特徴とするデータ処理方法。

【請求項22】 前記生成工程は、前記抽出工程で抽出したメタデータの全てに含まれる共通のメタデータ項目に基づいて前記ディレクトリに対応するメタデータを生成する

ことを特徴とする請求項21に記載のデータ処理方法。

【請求項23】 前記生成工程は、前記抽出工程で抽出したメタデータに最も多く含まれている共通のメタデータ項目に基づいて前記ディレクトリに対応するメタデータを生成する

ことを特徴とする請求項21に記載のデータ処理方法。

【請求項 2 4】 新たなディレクトリを生成し、前記生成工程において生成されたディレクトリ用のメタデータに用いられたメタデータ項目を含むメタデータが付加されているデータファイルを該新たなディレクトリに登録する第 1 登録工程を更に備え、

前記付加工程は、前記新たなディレクトリに対応するディレクトリデータに前記生成工程で生成されたメタデータを付加する

ことを特徴とする請求項 2 1 に記載のデータ処理方法。

【請求項 2 5】 新たな別のディレクトリを生成し、前記生成工程において生成されたディレクトリ用のメタデータに用いられたメタデータ項目が存在しないメタデータが付加されているデータファイルを該新たな別のディレクトリに登録する第 2 登録工程を更に備える

ことを特徴とする請求項 2 1 に記載のデータ処理方法。

【請求項 2 6】 前記データファイルは、画像データファイル、音声データファイル、動画データファイルのいずれかである

ことを特徴とする請求項 2 1 に記載のデータ処理方法。

【請求項 2 7】 前記付加工程は、ディレクトリデータの後ろに前記生成工程で生成されたメタデータを接続する

ことを特徴とする請求項 2 1 に記載のデータ処理方法。

【請求項 2 8】 前記メタデータが XML のデータ記述言語で記述されている

ことを特徴とする請求項 1 乃至 2 7 のいずれかに記載のデータ処理方法。

【請求項 2 9】 前記メタデータが SGML のデータ記述言語で記述されている

ことを特徴とする請求項 1 乃至 2 7 のいずれかに記載のデータ処理方法。

【請求項 3 0】 前記メタデータが HTML のデータ記述言語で記述されている

ことを特徴とする請求項 1 乃至 2 7 のいずれかに記載のデータ処理方法。

【請求項 3 1】 ディレクトリ構造でファイルを管理するファイル管理システムにおいてディレクトリに対応するディレクトリデータを読み込む第 1 読込手

段と、

前記ディレクトリデータに付与すべきメタデータを読み込む第2読込手段と、

前記第1読込手段で読み込まれたディレクトリデータの後に、前記第2読込手段で読み込まれたメタデータを接続する接続手段と、

前記接続手段によって得られたデータの全体をディレクトリデータファイルとして出力する出力手段と

を備えることを特徴とするデータ処理装置。

【請求項32】 前記第2読込手段で読み込まれたメタデータが、所定のデータ記述言語における適正な形式で記述されているか否かを判定する判定手段を更に備え、

前記接続手段は、前記判定手段で適正な形式で記述されていると判定された場合に、前記メタデータを前記ディレクトリデータの後に接続する

ことを特徴とする請求項31に記載のデータ処理装置。

【請求項33】 前記判定手段は、前記メタデータが前記所定のデータ記述言語としての正当性を満足するか否かに基づいて判定する

ことを特徴とする請求項32に記載のデータ処理装置。

【請求項34】 前記判定手段は、前記メタデータがデータ記述言語が文法的に正しいか否かに基づいて判定する

ことを特徴とする請求項32に記載のデータ処理装置。

【請求項35】 ディレクトリデータファイルのデータにメタデータが登録されているか否かを判別するデータ処理装置であって、

ディレクトリデータファイルを読み込む読込手段と、

前記読込手段で読み込まれたディレクトリデータファイルのデータを末尾より検査し、所定のデータ記述言語における適正な形式で記述されたデータが存在するか否かを判定することにより、該データに含まれるメタデータを判別する判別手段と

を備えることを特徴とするデータ処理装置。

【請求項36】 前記判別手段においてメタデータが判別された場合、判別されたメタデータを抽出して出力する出力手段を更に備える

ことを特徴とする請求項 3 5 に記載のデータ処理装置。

【請求項 3 7】 前記出力手段は、前記抽出されたメタデータに基づく表示を行う

ことを特徴とする請求項 3 6 に記載のデータ処理装置。

【請求項 3 8】 前記出力手段は、前記抽出されたメタデータを、前記所定のデータ記述言語に基づいて所定の処理を実行するためのツールに提供する

ことを特徴とする請求項 3 6 に記載のデータ処理装置。

【請求項 3 9】 前記判別手段は、

前記所定のデータ記述言語によって規定された末尾文字列が前記データの末尾に存在するか否かをチェックするチェック手段と、

該末尾文字列が存在する場合に前記所定のデータ記述言語に規定された先頭文字列を該データの先頭へ向かって検索する検索手段とを備え、

前記末尾文字列と前記先頭文字列によって挟まれたデータが存在する場合に、このデータをメタデータと判別する

ことを特徴とする請求項 3 5 に記載のデータ処理装置。

【請求項 4 0】 前記判別手段は、前記末尾文字列と前記先頭文字列によって挟まれたデータが、前記所定のデータ記述言語における適正な形式を有するか否かを検査する検査手段を更に備える

ことを特徴とする請求項 3 9 に記載のデータ処理装置。

【請求項 4 1】 前記検査手段は、前記所定のデータ記述言語としての正当性を満足するかの判断を含めて検査する

ことを特徴とする請求項 4 0 に記載のデータ処理装置。

【請求項 4 2】 前記検査手段は、前記所定のデータ記述言語が文法的に正しいか否かの判断を含めて検査する

ことを特徴とする請求項 4 0 に記載のデータ処理装置。

【請求項 4 3】 処理対象データが属するディレクトリに対応するディレクトリデータを読み込む読込手段と、

前記読込手段で読み込まれたディレクトリデータよりメタデータを抽出する抽出手段と、

前記抽出手段で抽出されたメタデータに基づいて、前記処理対象データへメタデータを付加する付加手段と

を備えることを特徴とするデータ処理装置。

【請求項 4 4】 前記付加手段は、

前記処理対象データにメタデータが登録されているか否かを判定する判定手段を備え、

メタデータが登録されていないと判定された場合に、前記処理対象データに前記抽出手段で抽出されたメタデータを付加する

ことを特徴とする請求項 4 3 に記載のデータ処理装置。

【請求項 4 5】 前記付加手段は、

前記処理対象データにメタデータが登録されているか否かを判定する判定手段と、

メタデータが登録されていると判定された場合に、前記処理対象データからメタデータを分離する分離手段と、

前記分離手段で得られたメタデータと、前記抽出手段で得られたメタデータとに基づいて新たなメタデータを生成する生成手段とを備え、

前記生成手段によって生成されたメタデータを、前記分離手段によってメタデータが分離された後の前記処理対象データに付加する

ことを特徴とする請求項 4 3 に記載のデータ処理装置。

【請求項 4 6】 前記生成手段は、前記分離手段で得られたメタデータと、前記抽出手段で得られたメタデータとに含まれる全てのデータ項目が含まれるように新たなメタデータを生成する

ことを特徴とする請求項 4 5 に記載のデータ処理装置。

【請求項 4 7】 前記処理対象データは、他のディレクトリにコピーされたデータである

ことを特徴とする請求項 4 3 に記載のデータ処理装置。

【請求項 4 8】 前記処理対象データは、他のディレクトリに移動されたデータである

ことを特徴とする請求項 4 3 に記載のデータ処理装置。

【請求項49】 前記処理対象データはバイナリデータ部分を含み、  
前記付加手段は、前記処理対象データのバイナリデータ部分の後にメタデータを接続する

ことを特徴とする請求項43に記載のデータ処理装置。

【請求項50】 前記処理対象データは、画像データ、音声データ、動画データ  
データのいずれかである

ことを特徴とする請求項43に記載のデータ処理装置。

【請求項51】 指定されたディレクトリ内に属するデータファイルを読み  
込む読込手段と、

前記読込手段で読み込まれたデータファイルからメタデータを抽出する抽出  
手段と、

前記抽出手段で抽出されたメタデータに基づいてディレクトリ用のメタデー  
タを生成する生成手段と、

前記生成手段で生成されたメタデータをディレクトリデータに付加する付加手  
段とを

備えることを特徴とするデータ処理装置。

【請求項52】 前記生成手段は、前記抽出手段で抽出したメタデータの全  
てに含まれる共通のメタデータ項目に基づいて前記ディレクトリに対応するメ  
タデータを生成する

ことを特徴とする請求項51に記載のデータ処理装置。

【請求項53】 前記生成手段は、前記抽出手段で抽出したメタデータに最  
も多く含まれている共通のメタデータ項目に基づいて前記ディレクトリに対応  
するメタデータを生成する

ことを特徴とする請求項51に記載のデータ処理装置。

【請求項54】 新たなディレクトリを生成し、前記生成手段において生成  
されたディレクトリ用のメタデータに用いられたメタデータ項目を含むメタデー  
タが付加されているデータファイルを該新たなディレクトリに登録する第1登  
録手段を更に備え、

前記付加手段は、前記新たなディレクトリに対応するディレクトリデータに前

記生成手段で生成されたメタデータを付加する

ことを特徴とする請求項 5 1 に記載のデータ処理装置。

【請求項 5 5】 新たな別のディレクトリを生成し、前記生成手段において生成されたディレクトリ用のメタデータに用いられたメタデータ項目が存在しないメタデータが付加されているデータファイルを該新たな別のディレクトリに登録する第 2 登録手段を更に備える

ことを特徴とする請求項 5 1 に記載のデータ処理装置。

【請求項 5 6】 前記データファイルは、画像データファイル、音声データファイル、動画データファイルのいずれかである

ことを特徴とする請求項 5 1 に記載のデータ処理装置。

【請求項 5 7】 前記付加手段は、ディレクトリデータの後ろに前記生成手段で生成されたメタデータを接続する

ことを特徴とする請求項 5 1 に記載のデータ処理装置。

【請求項 5 8】 前記メタデータが XML のデータ記述言語で記述されている

ことを特徴とする請求項 3 2 乃至 5 7 のいずれかに記載のデータ処理装置。

【請求項 5 9】 前記メタデータが SGML のデータ記述言語で記述されている

ことを特徴とする請求項 3 2 乃至 5 7 のいずれかに記載のデータ処理装置。

【請求項 6 0】 前記メタデータが HTML のデータ記述言語で記述されている

ことを特徴とする請求項 3 2 乃至 5 7 のいずれかに記載のデータ処理装置。

【請求項 6 1】 請求項 1 乃至 3 0 のいずれかに記載の方法をコンピュータによって実現するための制御プログラムを格納したことを特徴とする記憶媒体。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、データにメタデータを登録、判別することを可能とするデータ処理方法および装置に関する。

## 【 0 0 0 2 】

## 【従来の技術】

メタデータ (meta-data) とは、「データに関するデータ」であり、画像データや音声データ等のバイナリデータを説明するデータとして用いられている。しかし、バイナリデータとこれに対応するメタデータが別々のファイルで存在した場合、ファイルの移動やコピーの際に、ユーザはバイナリデータとメタデータとを同時に管理しなければならず、非常にわずらわしいことになる。

## 【 0 0 0 3 】

そこで一般に、バイナリデータとメタデータの管理を容易にするために、バイナリデータとメタデータを記述する様々な方法が提案されてきた。この種の従来技術は、新しいバイナリフォーマットを規定する方法と、データベースで管理する方法の2つに分けることができる。

## 【 0 0 0 4 】

まず、新しいバイナリフォーマットを規定する方法の一例をあげると、画像フォーマットではTiff、Exif、Flashpixなどがある。図16は、バイナリデータにメタデータを埋め込んだフォーマットの概観を示す図である。バイナリデータとしては、例えば画像データが挙げられる。図16に示されるように、画像のヘッダ部分にメタデータを記述する枠組みを設け、そこにユーザがメタデータを記述するというのが一般的な方法である。このようにメタデータを記述することにより、データの検索・分類が容易になる。また、バイナリデータ内にメタデータを含むようになるので、1つのファイルで管理でき、ファイルの管理は比較的容易になる。

## 【 0 0 0 5 】

次に、バイナリデータとメタデータをデータベースで管理する方法を説明する。図17はバイナリデータとメタデータをデータベースで管理する方法を概念的に示した図である。図17に示されるような、別々のファイルで存在するバイナリデータとメタデータをデータベース等を用いて管理するという方法も広く行われているものである。この場合は既存のバイナリデータが、既存のアプリケーションでそのまま使えるという利点がある。



## 【 0 0 0 6 】

また、ファイル管理においてディレクトリ構造が用いられ得ることはよく知られている。従って、例えば大量のバイナリ・データファイルを管理する場合は、通常1つのディレクトリの下に全てのバイナリ・データファイルを置くようなことはせず、あるまとまった属性や特性を持つバイナリ・データファイル群毎にサブディレクトリを作成し、その下に複数のバイナリ・データファイルを置くことが多い。この場合、1つのサブディレクトリの下に置かれている各バイナリ・データファイルのメタデータには共通の項目があることになる。しかしながら、現状では、各バイナリ・データファイルの各々が独立に、メタデータを上記2つの方式のいずれかの方法により管理していた。

## 【 0 0 0 7 】

## 【発明が解決しようとする課題】

上述したような、メタデータを記述する新フォーマットを規定する方法とデータベースを用いてメタデータを管理する方法のそれぞれに問題がある。

## 【 0 0 0 8 】

まず、メタデータを記述する新フォーマットを規定した場合には、既存のバイナリデータを当該新フォーマットに変換し、なおかつその新フォーマット内にメタデータを記述しなければならぬ。更に、その新フォーマット内のメタデータを用いて検索するためには、当該新フォーマット対応のアプリケーションが必要となる。すなわち、メタデータを記述したり利用したりするために、非常に多くのステップと専用の環境が必要になるという問題がある。また、このような新フォーマットのバイナリデータを処理する（例えば画像データであれば画像の再生）ためには、当該フォーマットに対応したアプリケーションが必要であり、既存のアプリケーションでは対応できなくなる。

## 【 0 0 0 9 】

そのうえ、メタデータの記述方法も新フォーマットにおいて独自に決められたものであり、新フォーマット内のメタデータを利用するアプリケーションを作成するためには、新規にメタデータの検索ルーチンをつくらなければならないという問題もある。さらに、新しい枠組みのメタデータを記述するにはフォーマット

の規定を変更しなければならないという問題点もあった。

【0010】

一方、データベースを用いてバイナリデータとメタデータを同時に管理する場合、データベースソフトが無ければメタデータの登録も利用もできないという問題があった。また、登録したメタデータを表示するためにも専用のソフトウェアが必要である。更に、バイナリデータをデータベース外に持っていくと、メタデータは付加されず、メタデータのないバイナリデータになってしまうという問題点もあった。

【0011】

更に、ディレクトリ構造でバイナリ・データファイルを管理する場合においても、1つのディレクトリの下に置かれているバイナリ・データファイルのメタデータは、各バイナリ・データファイル毎に存在し、それぞれ独立して管理される。このため、複数のバイナリ・データファイルに存在する共通項目についても各バイナリ・データファイル毎独立して管理することになり、管理するメタデータのデータ量が多くなってしまいう問題がある。また、その共通項目を変更する場合には、全てのバイナリ・データファイルのメタデータに対して逐一変更を行わなければならないという欠点もある。

【0012】

本発明はメタデータの記述・検索に関する上記の問題点に鑑みてなされたものであり、既存のアプリケーションに影響を与えずに、ディレクトリデータにメタデータを登録可能とすることを目的とする。

また、本発明の他の目的は、メタデータの記述に一般的なデータ記述言語を用いることにより、データ記述言語用の既存のツールを利用することを可能とし、対応アプリケーションの開発を容易にすることにある。

また、本発明の他の目的は、メタデータが記述されたディレクトリデータからメタデータを抽出し、例えば検索、参照、変更等の処理に供することを可能とすることにある。

また、本発明の他の目的は、ディレクトリに対してメタデータを登録することにより、メタデータの管理容易化、データ量の削減を図ることにある。

更に、本発明の他の目的は、ある属性あるいは特性などによりグループ化されたディレクトリにより管理されているデータファイルに対して簡便にメタデータを付属可能とすることにある。

#### 【 0 0 1 3 】

また、本発明の他の目的は、ディレクトリからファイルを移動したりコピーした場合であっても、ディレクトリに登録していたメタデータが失われてしまったり、登録したメタデータが無駄になってしまうことを防止することにある。

また、本発明の他の目的は、既に各バイナリデータにメタデータが登録されている場合に、それらメタデータをもとに適切なディレクトリ用のメタデータを生成、付加することを可能にすることにある。

また、本発明の他の目的は、メタデータが登録されたデータファイルから、適切なメタデータを抽出し、そのメタデータをディレクトリデータ登録することによって、検索を高速化することを目的とする。

更に、本発明の他の目的は、データファイルを、ある属性あるいは特性などによりグループ化したディレクトリに簡便に登録する可能とすることを目的とする。

#### 【 0 0 1 4 】

##### 【課題を解決するための手段】

上記の目的の少なくとも一つを達成するための本発明の一態様によるデータ処理方法は例えば以下の構成を備える。すなわち、

ディレクトリ構造でファイルを管理するファイル管理システムにおいてディレクトリに対応するディレクトリデータを読み込む第 1 読込工程と、

前記ディレクトリデータに付与すべきメタデータを読み込む第 2 読込工程と、

前記第 1 読込工程で読み込まれたディレクトリデータの後に、前記第 2 読込工程で読み込まれたメタデータを接続する接続工程と、

前記接続工程によって得られたデータの全体をディレクトリデータファイルとして出力する出力工程とを備える。

#### 【 0 0 1 5 】

また、上記の目的の少なくとも一つを達成するための本発明の他の態様による

データ処理方法は例えば以下の構成を備える。すなわち、

ディレクトリデータファイルのデータにメタデータが登録されているか否かを判別するデータ処理方法であって、

ディレクトリデータファイルを読み込む読込工程と、

前記読込工程で読み込まれたディレクトリデータファイルのデータを末尾より検査し、所定のデータ記述言語における適正な形式で記述されたデータが存在するか否かを判定することにより、該データに含まれるメタデータを判別する判別工程とを備える。

【 0 0 1 6 】

また、上記の目的の少なくとも一つを達成するための本発明の他の態様によるデータ処理方法は例えば以下の構成を備える。すなわち、

処理対象データが属するディレクトリに対応するディレクトリデータを読み込む読込工程と、

前記読込工程で読み込まれたディレクトリデータよりメタデータを抽出する抽出工程と、

前記抽出工程で抽出されたメタデータに基づいて、前記処理対象データへメタデータを付加する付加工程とを備える。

【 0 0 1 7 】

また、上記の目的の少なくとも一つを達成するための本発明の他の態様によるデータ処理方法は例えば以下の構成を備える。すなわち、

指定されたディレクトリ内に属するデータファイルを読み込む読込工程と、

前記読込工程で読み込まれたデータファイルからメタデータを抽出する抽出工程と、

前記抽出工程で抽出されたメタデータに基づいてディレクトリ用のメタデータを生成する生成工程と、

前記生成工程で生成されたメタデータをディレクトリデータに付加する付加工程とを備える。

【 0 0 1 8 】

また、本発明の他の態様によれば、上記のデータ処理方法を実現するデータ処

理装置が提供される。さらに本発明の他の態様によれば、上記データ処理方法をコンピュータに実現させるためのコンピュータプログラムを格納した記憶媒体が提供される。

【 0 0 1 9 】

【発明の実施の形態】

以下、添付の図面を参照して本発明の好適な実施形態を説明する。

【 0 0 2 0 】

<第 1 の実施形態>

図 1 は第 1 の実施形態によるデータ処理装置の構成を示すブロック図である。図 1 において、100 は読込部であり、スキャナ装置などを用いて画像を読み込む。101 は入力部であり、ユーザからの指示やデータを入力するもので、キーボードやポインティング装置を含む。102 は蓄積部であり、バイナリデータやメタデータをディレクトリ構造で蓄積する。蓄積部 102 としては、ハードディスクを用いるのが一般的である。103 は表示部であり、蓄積部 102 に蓄積されたバイナリデータを表示したり、読込部 100 で読み込まれた画像データを表示する。表示部 103 としては、CRT や液晶表示装置が一般的である。

【 0 0 2 1 】

104 は CPU であり、上述した各構成の処理のすべてに関わり、ROM 105 と RAM 106 はその処理に必要なプログラム、データ、或いは作業領域を CPU 104 に提供する。なお、図 2 のフローチャートを参照して後述する本実施形態の処理手順を実現するための制御プログラムも ROM 105 に格納されているものとする。もちろん、蓄積部 102 にその制御プログラムを格納しておき、CPU 104 による実行に応じてその制御プログラムが RAM 106 上へロードされるような構成であってもよい。

【 0 0 2 2 】

なお、第 1 の実施形態のデータ処理装置には上記以外にも、種々の構成要素が設けられているが、本発明の主眼ではないので、その説明については省略する。

【 0 0 2 3 】

つぎに、以上のように構成されたデータ処理装置において、メタデータをディ

レクトリデータに登録する処理について説明する。図 2 は、第 1 の実施形態によるメタデータの登録処理を説明するフローチャートである。

【 0 0 2 4 】

図 2 において、まず、ステップ S 3 0 1 で、ユーザによって指定されたディレクトリのディレクトリデータをメモリ（RAM 1 0 6）上に読み込む。これは例えば所望のディレクトリ名をキーボードから入力したり、ポインティング装置（例えばマウス）によって当該ディレクトリを指示することによりなされる。次にステップ S 3 0 2 において、ユーザによって指定された、メタデータが記述されている XML ファイルをメモリ（RAM 1 0 6）上に読み込む。この XML ファイルの指定も、キーボードからファイル名を入力したり、ポインティング装置（例えばマウス）で対応するアイコンを指示する等によって行われる。

【 0 0 2 5 】

次にステップ S 3 0 3 で、メタデータを記述した XML ファイルが適正形式の XML データであるかを調べる。この適性形式の判定では、XML ファイルの記述フォーマットを満足しているか（例えば、タグの左右の括弧が正しく対をなしているか、タグ付けの形式が正しいかどうか等の文法的な正しさ）がチェックされる。なお、適性形式の XML データであるか否かの判定は、正当な XML データであるか否かを含めたチェックであってもよい。ここで、正当な XML データか否かの判定は、例えば、XML データが DTD（Document Type Definition）等のスキーマに従って記述されているかどうか等のチェックを行うことでなされる。

【 0 0 2 6 】

ステップ S 3 0 3 において適正形式の XML データでないと判定された場合にはステップ S 3 0 5 に進む。ステップ S 3 0 5 では、XML データにエラーがある旨を表示部 1 0 3 に表示し、本処理を終了する。

【 0 0 2 7 】

一方、ステップ S 3 0 3 において XML ファイルが適正形式の XML データであると判定された場合には、処理はステップ S 3 0 4 に進む。ステップ S 3 0 4 では、ステップ S 3 0 1 でメモリ上に読み込まれたディレクトリデータの後ろに

当該メタデータを接続することにより、メタデータの登録を行う。その後、ステップ S 3 0 6 において、メタデータを登録したディレクトリデータを出力し、処理を終了する。なお、ステップ S 3 0 6 におけるデータ出力により、図 3 に示されるデータ構造を有するメタデータ付ディレクトリデータが 1 つのディレクトリデータファイルとして蓄積部 1 0 2 に格納されることになる。

## 【 0 0 2 8 】

図 3 は本実施形態によるディレクトリデータへのメタデータの登録状態を説明する図である。図 3 に示されるように、ディレクトリデータの最後に（本例では、ディレクトリデータの終端を示す識別子〈E O F〉の後に）、XML データで記述されたメタデータを接続する。このようにすることにより、他のアプリケーションには影響を与えずに、メタデータをディレクトリデータに登録することができる。すなわち、一般のアプリケーションはディレクトリデータを参照する場合に、ディレクトリデータの先頭から終端識別子までのデータを用いるので、接続されたメタデータは何等影響を及ぼさないのである。

## 【 0 0 2 9 】

さらに、メタデータは XML で記述されているため、この XML データ部分を抽出しておくことにより、XML データを理解するツールがあれば、メタデータの追加・変更・参照が可能であり、非常に汎用性に優れている。なお、ディレクトリデータから XML データ部分を抽出する処理については第 2 の実施形態で詳しく説明する。

## 【 0 0 3 0 】

以上説明したように、第一の実施形態によれば、メタデータを XML で記述し、ディレクトリデータの最後に接続することにより、既存のアプリケーションに影響を及ぼさずに、既存のディレクトリデータにメタデータを登録することができる。

## 【 0 0 3 1 】

また、ディレクトリデータにメタデータを登録することによって、当該ディレクトリ下のすべてのファイルにメタデータをつけることが不要となり、その内容の変更も簡単になる。

## 【0032】

## ＜第2の実施形態＞

第1の実施形態においてディレクトリデータにメタデータを登録して、ディレクトリデータファイルとする方法を説明した。第2の実施形態では、ディレクトリデータファイルにメタデータが登録されているかどうかを判別し、登録されている場合にはそのメタデータを抽出する処理について説明する。なお、第2の実施形態におけるデータ処理装置の構成は第1の実施形態（図1）と同様であるのでここでは説明を省略する。

## 【0033】

以下、指定されたディレクトリのディレクトリデータファイルに第1の実施形態で説明した如きメタデータが登録されているか否かの判定と、登録されたメタデータを抽出する動作について説明する。図4は第2の実施形態による登録されたメタデータの判別及び抽出手順を示すフローチャートである。なお、本実施形態では、抽出されたメタデータを表示部103に表示するが、出力の形態はこれに限らない。例えば、抽出したメタデータを編集や検索等の処理に提供するように構成してもよいことは当業者には明らかであろう。

## 【0034】

図4によれば、まず、ステップS501で、ユーザの指示により、メタデータが登録されているかを判別したいディレクトリデータファイル、即ち処理対象データを指定する。ステップS501における、処理対象データの指定は、キーボードから当該ディレクトリデータのファイル名を入力したり、対応するアイコンをポインティング装置（マウス）で指示することにより行われる。

## 【0035】

次にステップS502において、指定されたディレクトリデータファイルのデータにXMLで記述されたメタデータが登録されているかどうかを判別する。以下、ステップS502における判別処理の詳細について図5のフローチャートと、図6の概略図にしたがって説明する。図5は第2の実施形態によるメタデータの判別処理の詳細を説明するフローチャートである。また、図6はメタデータとしてXMLデータが登録されたディレクトリデータのデータ構成例を示す図であ



る。

【 0 0 3 6 】

第 1 の実施形態で説明したように、メタデータとしての XML データが登録されているディレクトリデータファイル（処理対象データ）のデータ構成は図 6 のようになっている。したがって、メタデータの有無の判別は以下のように行われる。

【 0 0 3 7 】

図 5 に示されるように、まず、ステップ S 6 0 1 で、上記ステップ S 5 0 1 で指定されたディレクトリデータファイルのデータ全体（処理対象データの全体）をメモリ（RAM 1 0 6）上に読み込む。なお、第 1 の実施形態のステップ S 3 0 6 によって出力されたデータは一つのディレクトリデータファイルとして管理されるので、ファイル管理システムによってこのデータの全体を読み出すことが可能である。

【 0 0 3 8 】

次にステップ S 6 0 2 において、ステップ S 6 0 1 で読み込んだデータの最後に“</PhotoXML>”という文字列があるか調べる。存在しなかった場合はステップ S 6 0 5 に進む。

【 0 0 3 9 】

一方、読み込んだ処理対象データの最後に、“</PhotoXML>”という文字列が存在した場合はステップ S 6 0 3 にすすむ。ステップ S 6 0 3 では“</PhotoXML>”という文字列の前に“<PhotoXML>”という文字列が存在するかどうかを調べる。

【 0 0 4 0 】

ステップ S 6 0 3 において“<PhotoXML>”という文字列の存在が確認された場合は、ステップ S 6 0 4 にすすむ。ステップ S 6 0 4 においては、メタデータが登録されているものと結論づけ、本処理を終了する。一方、ステップ S 6 0 3 において“<PhotoXML>”という文字列の存在が確認されなかった場合には、処理はステップ S 6 0 5 に進む。ステップ S 6 0 5 においては、メタデータは登録されていないものと結論づける。すなわち、ステップ S 6 0 2 で、当該バイナリ

データの最後に文字列“</PhotoXML>”が存在しない場合、ステップS 6 0 3で文字列“<PhotoXML>”が存在しない場合には、処理はステップS 6 0 5に進み、当該処理対象データにメタデータは登録されていないものと結論づける。

【 0 0 4 1 】

なお、ステップS 6 0 3において、“<PhotoXML>”という文字列の存在が確認された後に、さらにそれらの文字列で囲まれたデータが、XMLの適正形式で記述されているかを確認するようにしてもよい。更に、この場合、当該データがXMLの正当なデータであるか否かの判定を含めて行うようにしてもよい。適性形式か否かの判定、正当なデータか否かの判定は、第1の実施形態（ステップS 3 0 3）で説明したとおりである。

【 0 0 4 2 】

次に、図4のフローチャートにもどる。図5のフローチャートで示される処理によってメタデータが登録されていると結論づけられた場合には、処理はステップS 5 0 3に進む。ステップS 5 0 3では、文字列“<PhotoXML>”と“</PhotoXML>”で囲まれた部分のXMLデータに基づいて登録されているメタデータの内容を表示部1 0 3に表示し、処理を終了する。一方、ステップS 5 0 2でメタデータが登録されていないと判定された場合にはそのまま処理を終了する。

【 0 0 4 3 】

以上説明したように、第2の実施形態によれば、メタデータを登録したディレクトリデータと、通常のディレクトリデータとを、XMLの記述規則によって判別し、メタデータが登録されているディレクトリデータの場合には、そのメタデータを表示することができる。

【 0 0 4 4 】

すなわち、第2の実施形態によれば、メタデータが登録されたディレクトリデータとメタデータが登録されていないディレクトリデータとを判別するとともに、登録されたメタデータを抽出することが可能となる。従って、メタデータとして既存のデータ記述言語を用いれば、メタデータの参照や編集、検索に際して、当該データ記述言語用の既存のツールをそのまま用いることができ、開発に関する手間も省くことができる。

## 【0045】

次に図7を用いて具体的にディレクトリ毎に格納されているバイナリ・データファイルのメタデータを格納する方法について説明する。説明を簡単にするため、ルートディレクトリDirectory 1の下に、Directory 2, Directory 3, Directory 4の3つのサブディレクトリがあり、各サブディレクトリの下に図7に示すようにバイナリ・データファイルが置かれているものとする。

## 【0046】

このような場合において、各ディレクトリのディレクトリデータにメタデータを登録すれば、Directory 2の下に置かれているBinary Data File 2-1からBinary Data File 2-3には同一のメタデータが、Directory 3の下のBinary Data File 3-1 とBinary Data File 3-2にも同一のメタデータが、同様にDirectory 4の下に置かれているファイルにも同一のメタデータが夫々保存されることになる。

## 【0047】

メタデータの簡単な例を示すと、例えばデジタルカメラで撮影した場合などでは、Directory 2は、3月10日に撮影し、Directory 3のバイナリデータは3月11日に撮影したといった情報である。あるいは、「風景」や「人物」等の属性でディレクトリを分けることも考えられる。いずれにせよ、サブディレクトリを作成する時のデータファイルの分類の仕方によりメタデータに格納する項目や内容を決めることができる。

## 【0048】

上記実施形態の場合、各バイナリ・データファイル毎にメタデータを保存するのではなく、各サブディレクトリのディレクトリデータにメタデータが格納される。従ってバイナリ・データファイルを読み出そうとするアプリケーションは、そのファイルが属しているディレクトリに付随するメタデータを読み出し、これをそのバイナリ・データファイルのメタデータとして処理するように動作する。

## 【0049】

例えば、メタデータに格納されているある項目でバイナリ・データファイルを検索する場合は、本実施形態によれば、ディレクトリデータに付随して格納されているメタデータがディレクトリの下に置かれているバイナリ・データファイル

全てに対して有効なため、ディレクトリのみを検索すればよく、高速に検索処理を行うことが可能となる。

【 0 0 5 0 】

なお、ディレクトリに格納されるデータをバイナリ・データとして説明したが、いかなる種類のデータであってもよいことは明らかである。

【 0 0 5 1 】

＜ファイルシステムによるディレクトリデータの更新について＞

ところで、ディレクトリに対してファイルの追加や削除を行った場合、ファイルシステムはこの処理に応じてディレクトリデータを更新する。しかしながら、メタデータが接続された状態のままで更新を行うと、接続されているメタデータが破損してしまう。従って、上記第1の実施形態によってメタデータが付加されたディレクトリデータを扱うために、ファイルシステムに若干の変更が必要となる。以下、ファイル管理システムによるディレクトリデータの更新処理について説明する。

【 0 0 5 2 】

図8は、本実施形態のファイル管理システムによるディレクトリデータの更新処理を説明するフローチャートである。ディレクトリ内のファイルが更新、追加、削除された場合は、図8に示す処理によってディレクトリデータの更新が行われる。

【 0 0 5 3 】

まず、ステップS701において、当該ディレクトリのディレクトリデータファイルが読み出される。そして、ステップS702において、当該ディレクトリデータにメタデータが接続されているか否かを判定する。この判定は、上述の図5に示される手順により行うことができる。そして、メタデータが登録されていると判定された場合は、ステップS703へ進む。

【 0 0 5 4 】

ステップS703では、存在が検出されたメタデータを分離し、別の記憶エリアに待避させておく。そして、ステップS704では、メタデータが分離された後のディレクトリデータに対して、従前と同様の方法で編集を施す。そして、デ



ィレクトリデータの編集を終えたならば、ステップS705において、先のステップS703で分離し待避させておいたメタデータを、更新後のディレクトリデータの後に再度接続する。一方、ステップS702においてメタデータが登録されていなければ、ステップS706へ進み、即座にディレクトリデータを更新する。

#### 【0055】

以上の処理において、ステップS704とステップS706におけるディレクトリデータの更新処理は、従前のファイル管理システムにおいて実行される処理と同一のものでよい。従って、上述のファイル管理システムの改造は大した作業ではなく、僅かな変更によって本実施形態のディレクトリファイルを取り扱い可能となることが当業者には理解されよう。

#### 【0056】

以上第1及び第2の実施形態で説明したように、メタデータをディレクトリデータに登録するのでディレクトリ単位でメタデータを登録、管理できる。

また、既存のディレクトリデータの最後にメタデータを接続することにより、既存のアプリケーションに影響を与えずに、ディレクトリデータにメタデータを登録することが可能となる。

更に、メタデータが記述されたディレクトリデータからメタデータを抽出し、例えば検索、参照、変更等の処理に供することが可能となる。従って、メタデータの記述に一般的なデータ記述言語を用いることにより、データ記述言語用の既存のツールを利用することが可能となり、対応アプリケーションの開発が容易である。

また、ディレクトリに対してメタデータを登録することにより、メタデータの管理容易化、データ量の削減が図られる。このため、例えば、メタデータ内のある項目でバイナリデータを検索する場合には、ディレクトリのみ検索を行えばよく、高速に検索することができる。また、メタデータ内の項目に格納されている値を変更する場合には、各バイナリデータファイル毎に行う必要が無く、ディレクトリデータに付属しているメタデータを変更するだけで、そのディレクトリで管理されているバイナリデータ全てに対して変更したことになる。このため、メ



タデータの変更を高速に行える。また、ディレクトリで管理されている複数のバイナリデータに対して共通なメタデータを一つのメタデータで管理することができ、データ量を削減することができる。

#### 【 0 0 5 7 】

##### ＜第3の実施形態＞

第1及び第2の実施形態ではディレクトリデータにメタデータを登録することを示した。ディレクトリにメタデータを登録することによって、当該ディレクトリ下のすべてのファイルにメタデータをつけることが不要となり、その内容の変更も簡単になった。しかしながら、このディレクトリからファイルを移動したりコピーした場合、ディレクトリに登録していたメタデータは失われてしまい、登録したメタデータが無駄になってしまう可能性がある。第3の実施形態では、このような不具合を解消するべく、ディレクトリ内のバイナリデータがディレクトリ外に複製・移動された場合に、そのバイナリデータに自動的にメタデータを登録する。なお、第3の実施形態におけるデータ処理装置の構成は第1の実施形態（図1）と同様であるのでここでは説明を省略する。

#### 【 0 0 5 8 】

次に、第3の実施形態によるバイナリデータへのメタデータの登録方法を図9のフローチャートを参照して説明する。なお、図9のフローチャートでは、ディレクトリ内のバイナリデータがディレクトリ外に複製・移動された場合に、そのバイナリデータにメタデータを自動的に付加する。

#### 【 0 0 5 9 】

まず、ステップS1301において、ユーザの指示でディレクトリ内の1つのバイナリデータが、他のディレクトリへの複製または移動のために指定される。なお、以下の処理は、操作内容が複製であった場合は、複製後のバイナリデータ（複製元のバイナリデータではない）に対して、操作内容が移動であった場合は、移動後のバイナリデータに対して実行される。次にステップS1302において、指定されたバイナリデータが属するディレクトリに、第1、第2の実施形態で説明したようなディレクトリメタデータが登録されているかを判別する。ディレクトリメタデータが登録されているか否かの判別は図5及び図6を参照して第

2の実施形態で上述したとおりである。

【0060】

ステップS1302による判別の結果、ディレクトリにディレクトリデータが登録されている場合はステップS1303にすすみ、登録されていない場合はステップS1304にすすむ。ステップS1304では、ディレクトリデータは登録されていないので、指定されたバイナリデータを複製（またはコピー）して処理を終える。

【0061】

一方、ディレクトリメタデータが登録されている場合は、ステップS1303において、ステップS1301で指定されたバイナリデータにメタデータが登録されているかどうかを判別する。バイナリデータにメタデータが登録されているか否かを判別する処理について、図10及び図11を用いて説明する。図10は、バイナリデータにメタデータが登録されているか否かを判定するためのフローチャートである。図11は本実施形態によるバイナリデータへのメタデータの登録状態を説明する図である。

【0062】

メタデータが登録されているバイナリデータの内部は図11のようになっている。すなわち、上述のディレクトリデータへのメタデータの付加と同様に、バイナリデータの後尾にXML形式で記述されたメタデータが接続されている。したがって、メタデータの有無の判別は、図10において、まずステップS1601で、指定されたバイナリデータの全部をメモリ上に読み込む。次にステップS1602において、読み込んだバイナリデータの最後に“</PhotoXML>”という文字列があるか調べる。もし存在した場合は、ステップS1603にすすむ。ステップS1603において、“</PhotoXML>”という文字列の前に“<PhotoXML>”という文字列が存在するか調べる。もし存在した場合は、ステップS1604に進み、当該バイナリデータにメタデータが登録されているものと結論づける。

一方、ステップS1602で“</PhotoXML>”という文字列が存在しないと判断された場合、或いはステップS1603で“<PhotoXML>”という文字列が

存在しないと判断された場合は、ステップ S 1 6 0 5 にすすみ、当該バイナリデータにメタデータは登録されていないと結論づける。

【 0 0 6 3 】

なお、ステップ S 1 6 0 3 において、“<PhotoXML>”という文字列の存在が確認された後に、さらにそれらの文字列で囲まれたデータが、XMLの適正形式で記述されているかを確認するようにしてもよい。更に、この場合、当該データがXMLの正当なデータであるか否かの判定を含めて行うようにしてもよい。適性形式か否かの判定、正当なデータか否かの判定は、第1の実施形態（ステップ S 3 0 3）で説明したとおりである。

【 0 0 6 4 】

以上の判別処理によりバイナリデータにメタデータが登録されていないと判別された場合にはステップ S 1 3 0 8 にすすむ。ステップ S 1 3 0 8 に処理が進んだ場合、ディレクトリにはディレクトリメタデータが登録され、バイナリデータにはメタデータが登録されていない状態である。従って、ディレクトリメタデータをバイナリデータに登録し、バイナリデータを複製(移動)して処理を終了する。なお、バイナリデータへのメタデータの登録は、第1の実施形態で説明したディレクトリデータへのメタデータの登録と同様の方法で行われる。すなわち、バイナリデータの末尾にメタデータを接続して、図 1 1 に示すような形態のデータを生成する。

【 0 0 6 5 】

一方、ステップ S 1 3 0 3 においてバイナリデータにメタデータが登録されていると判別された場合は、ステップ S 1 3 0 5 へ進む。ステップ S 1 3 0 5 においては、ディレクトリに登録されていたディレクトリメタデータとバイナリデータに登録されていたメタデータとを比較する。次に、ステップ S 1 3 0 6 において、ステップ S 1 3 0 5 で比較した結果、ディレクトリメタデータとメタデータすべてが含まれるようなメタデータを作成する。

【 0 0 6 6 】

ステップ S 1 3 0 6 におけるメタデータの生成処理について詳しく説明する。ステップ S 1 3 0 6 では、図 1 2 に示されるように、ディレクトリメタデータ 1



801とメタデータ1802をあわせて、それらの内容すべてが含まれるようなメタデータ1803を作成する。

【0067】

以上のようにして作成したメタデータをバイナリデータに新たに登録し、バイナリデータを複製(移動)して、処理を終了する。

【0068】

以上説明したように、第3の実施形態によればディレクトリに登録されたディレクトリメタデータを、ディレクトリ内のバイナリデータの複製・移動の際にメタデータとして登録することによって、ディレクトリ内のファイル(例えばバイナリデータ)が別の場所に移動しても、当該バイナリデータに対応するメタデータが失われることはない。

【0069】

また、あらかじめメタデータがバイナリデータに登録してあった場合でも、ディレクトリメタデータと合成して、ディレクトリとバイナリデータ両方のメタデータを登録することができる。

【0070】

さらに、メタデータはデータ記述言語で記述されているので、既存のデータ記述言語専用のツールをそのまま用いることができ、開発に関する手間も省くことができる。

【0071】

<第4の実施形態>

第4の実施形態では、あらかじめバイナリデータがメタデータを有しているような場合に、ディレクトリデータに適切なメタデータを登録可能とする。なお、第4の実施形態におけるデータ処理装置の構成は第1の実施形態(図1)と同様であるのでここでは説明を省略する。以下、ディレクトリ内のバイナリデータから適切なディレクトリ用のメタデータを生成する処理について、図13のフローチャートにしたがって説明する。

【0072】

図13において、まず、ステップS2301において、ユーザの指示で所望の

ディレクトリ（メタデータを生成、付加すべきディレクトリ）が指定される。次にステップS2302において、ディレクトリ内のデータファイルの一つを選択し、そのデータファイルにメタデータが登録されているかを判別する。データファイルにメタデータが登録されているか否かの判別手順については、図10および図11を用いて上述したとおりである。

#### 【0073】

バイナリデータにメタデータが登録されている場合には、ステップS2303にすすみ、メタデータが登録されているデータファイル名とメタデータをメモリ上のリスト1に追加する。一方、ステップS2302において、メタデータが登録されていない場合にはステップS2304にすすむ。ステップS2304においては、メタデータが登録されていないデータファイル名をメモリ上のリスト2に追加する。以上の処理を当該ディレクトリ内のすべてのファイルについて実行する（ステップS2305）。

#### 【0074】

次に、ステップS2306において、メタデータとメタデータが登録されているデータファイル名が記録されているリスト1からディレクトリメタデータを生成する。リストからディレクトリメタデータを生成する処理について図14および図15を参照して説明する。図14は、ステップS2306におけるディレクトリメタデータの生成手順を説明するフローチャートである。また、図15はステップS2306におけるディレクトリメタデータの生成を説明する図である。

#### 【0075】

ステップS2401において、リスト1に登録されたすべてのメタデータに共通のメタデータ項目があるかを調べる。ここで、すべてのメタデータに共通のメタデータ項目があった場合、ステップS2402にすすみ、すべてのメタデータに共通のメタデータ項目を抽出し、抽出されたメタデータ項目を用いてディレクトリメタデータを生成して処理を終了する。この処理を図15を用いて説明する。リスト1（501）によれば、img001.jpg、img002.jpg、img003.jpgの3つのイメージファイルにメタデータが付与されているのがわかる（ファイル名は<Filename>と</Filename>で囲まれた文字列であり、メタデータは<PhotoXML>

と</PhotoXML>で囲まれたデータである)。リスト1(501)にリストされている3つのデータファイルのメタデータから共通のメタデータ項目を抽出してディレクトリメタデータを生成すると、ディレクトリメタデータ(502)のようになる。

#### 【0076】

一方、ステップS2401において全てのメタデータに共通のメタデータ項目が無かった場合には、ステップS2403において、最も多くのメタデータに登録されているメタデータ項目を用いてディレクトリメタデータを生成する。さらにステップS2404において、作成されたディレクトリメタデータに含まれるメタデータ項目を含まないメタデータが付加されているデータファイル名を掲載するリスト3を新たに生成する。そして、ステップS2405において、リスト3に掲載したデータファイル名とそのメタデータをリスト1から削除する。

#### 【0077】

図13に戻り、ステップS2307において、新しいディレクトリを生成し、リスト1に登録されているデータファイル(生成されたディレクトリ用のメタデータに含まれるメタデータ項目と共通するメタデータ項目を少なくとも1つ含むメタデータが登録されているデータファイル)をすべてこの新しいディレクトリに移動する。なお、ステップS2307において、新しいディレクトリはディレクトリメタデータからの適当なデータを用いて名前をつける。そして、ステップS2308において、上記ステップS2404でリスト3が生成されたか否かを判定し、生成されていればステップS2309へ進み、生成されていなければステップS2309をスキップする。ステップS2309では、リスト3にあるデータファイルを“MetaMisc”という名前のディレクトリに移動する。最後に、ステップS2310において、ステップS2307で生成した新しいディレクトリにディレクトリメタデータを登録して、処理を終了する。なお、ディレクトリにメタデータを登録する方法は第1の実施形態で説明したとおりであり、図6に示したようなデータ構造となる。

#### 【0078】

以上のようにしてディレクトリ内に含まれるデータファイルから、ディレクト

リメタデータを自動生成して、ディレクトリにメタデータを登録することができる。

【0079】

以上説明したように、第4の実施形態によればディレクトリ内のデータファイルからディレクトリメタデータを自動生成することが可能となり、容易にディレクトリにディレクトリメタデータを登録できる。共通のメタデータが無かった場合でも、“MetaMisc”内のバイナリデータから、ディレクトリメタデータを再度自動生成することで、ディレクトリメタデータを次々に生成していくことも可能である。

【0080】

なお、第4の実施形態において、指定されたディレクトリ内の全てのデータファイルが最後までリスト1に属する場合（これは、当該ディレクトリ内の全てのデータファイルにメタデータが登録されており、それら全てのメタデータが共通のメタデータ項目を有する場合である）には、新たなディレクトリを生成せず、当該指定されたディレクトリのディレクトリデータに生成されたメタデータを登録するようにしてもよい。

【0081】

以上の第4の実施形態の説明から明らかなように、バイナリデータからディレクトリメタデータを自動生成することが可能となる。また、ディレクトリにメタデータをつけることにより、高速な検索を行うことが可能となる。さらに、メタデータは既存のデータ記述言語で記述されているので、既存のデータ記述言語専用のツールをそのまま用いることができ、開発に関する手間も省くことができる。

【0082】

なお、上記第1～第4の各実施形態では、メタデータとしてXMLデータを用いたがこれに限られるものではない。例えば、SGMLやHTML等のデータ記述言語であってもよい。また、各ディレクトリに保持されるファイルのデータとしては、静止画像データ、動画データ、音声データ等のバイナリデータや、他のいかなる形態のデータであったもよい。

## 【 0 0 8 3 】

なお、本発明は、複数の機器（例えばホストコンピュータ、インタフェイス機器、リーダ、プリンタなど）から構成されるシステムに適用しても、一つの機器からなる装置（例えば、複写機、ファクシミリ装置など）に適用してもよい。

## 【 0 0 8 4 】

また、本発明の目的は、前述した実施形態の機能を実現するソフトウェアのプログラムコードを記録した記憶媒体を、システムあるいは装置に供給し、そのシステムあるいは装置のコンピュータ（またはCPUやMPU）が記憶媒体に格納されたプログラムコードを読み出し実行することによっても、達成されることは言うまでもない。

## 【 0 0 8 5 】

この場合、記憶媒体から読出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。

## 【 0 0 8 6 】

プログラムコードを供給するための記憶媒体としては、例えば、フロッピディスク、ハードディスク、光ディスク、光磁気ディスク、CD-ROM、CD-R、磁気テープ、不揮発性のメモリカード、ROMなどを用いることができる。

## 【 0 0 8 7 】

また、コンピュータが読出したプログラムコードを実行することにより、前述した実施形態の機能が実現されるだけでなく、そのプログラムコードの指示に基づき、コンピュータ上で稼働しているOS（オペレーティングシステム）などが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

## 【 0 0 8 8 】

さらに、記憶媒体から読出されたプログラムコードが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書込まれた後、そのプログラムコードの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるCPUなどが実際の処理の一部または全部を行い、

その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

## 【 0 0 8 9 】

## 【発明の効果】

以上説明したように、本発明によれば、メタデータをディレクトリデータに登録するのでディレクトリ単位でメタデータを登録、管理できる。

また、本発明によれば、既存のディレクトリデータの最後にメタデータを接続することにより、既存のアプリケーションに影響を与えずに、ディレクトリデータにメタデータを登録することが可能となる。

更に、本発明によれば、メタデータが記述されたディレクトリデータからメタデータを抽出し、例えば検索、参照、変更等の処理に供することが可能となる。従って、メタデータの記述に一般的なデータ記述言語を用いることにより、データ記述言語用の既存のツールを利用することが可能となり、対応アプリケーションの開発が容易である。

また、本発明によれば、ディレクトリに対してメタデータを登録することにより、メタデータの管理容易化、データ量の削減が図られる。このため、例えば、メタデータ内のある項目でバイナリデータを検索する場合には、ディレクトリのみ検索を行えばよく、高速に検索することができる。また、メタデータ内の項目に格納されている値を変更する場合には、各バイナリデータファイル毎行う必要が無く、ディレクトリデータに付属しているメタデータを変更するだけで、そのディレクトリで管理されているバイナリデータ全てに対して変更したことになり、高速に行える。また、ディレクトリで管理されている複数のバイナリデータに対して共通なメタデータを一つのメタデータで管理することができ、データ量を削減することができる。

## 【 0 0 9 0 】

また、本発明によれば、ディレクトリからファイルを移動したりコピーした場合であっても、ディレクトリに登録していたメタデータが失われてしまったり、登録したメタデータが無駄になってしまうことを防止できる。

また、本発明によれば、既に各バイナリデータにメタデータが登録されている

場合であっても、ディレクトリに適切なメタデータを付加することが可能になる。  
。

また、本発明によれば、メタデータが登録されたデータファイルから、適切なメタデータを抽出し、そのメタデータをディレクトリデータ登録することによって、検索を高速化することができる。

更に、本発明によれば、データファイルを、ある属性あるいは特性などによりグループ化したディレクトリに簡便に登録することができる。

【図面の簡単な説明】

【図 1】

第 1 の実施形態によるデータ処理装置の構成を示すブロック図である。

【図 2】

第 1 の実施形態によるメタデータの登録処理を説明するフローチャートである。  
。

【図 3】

実施形態によるディレクトリデータへのメタデータの登録状態を説明する図である。

【図 4】

第 2 の実施形態による登録されたメタデータの判別及び抽出手順を示すフローチャートである。

【図 5】

第 2 の実施形態によるメタデータの判別処理の詳細を説明するフローチャートである。

【図 6】

メタデータとしてXMLデータが登録されたディレクトリデータのデータ構成例を示す図である。

【図 7】

第 2 の実施形態によるディレクトリ構造の例を説明する図である。

【図 8】

第 2 の実施形態のファイル管理システムによるディレクトリデータの更新処理

を説明するフローチャートである。

【図 9】

第 3 の実施形態におけるバイナリデータへのメタデータ登録方法を示すフローチャートである。

【図 10】

第 3 の実施形態におけるバイナリデータにメタデータが登録されているかを判断する方法を示すフローチャートである。

【図 11】

第 3 の実施形態における、メタデータの登録されているバイナリデータのデータ構成例を示す図である。

【図 12】

第 3 の実施形態におけるメタデータとディレクトリメタデータを比較し合成する方法の例を説明する図である。

【図 13】

第 4 の実施形態における、ディレクトリデータに自動生成したディレクトリメタデータを登録する方法を示すフローチャートである。

【図 14】

第 4 の実施形態における、ディレクトリメタデータを自動生成する方法を示すフローチャートである。

【図 15】

第 4 の実施形態における、メタデータのリストからディレクトリメタデータを自動生成する例を示す図である。

【図 16】

バイナリデータにメタデータを埋め込んだフォーマットの概要を示す図である。

。

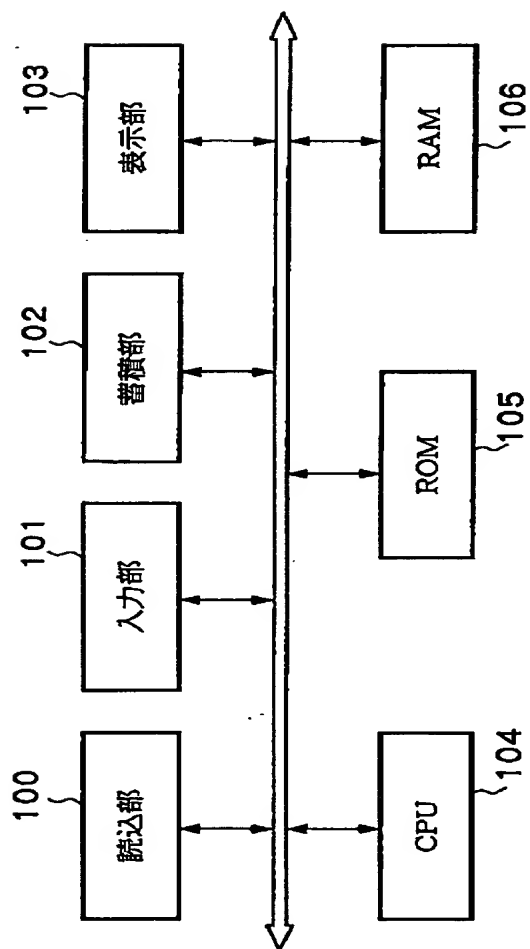
【図 17】

バイナリデータとメタデータをデータベースで管理する方法を概念的に示した図である。

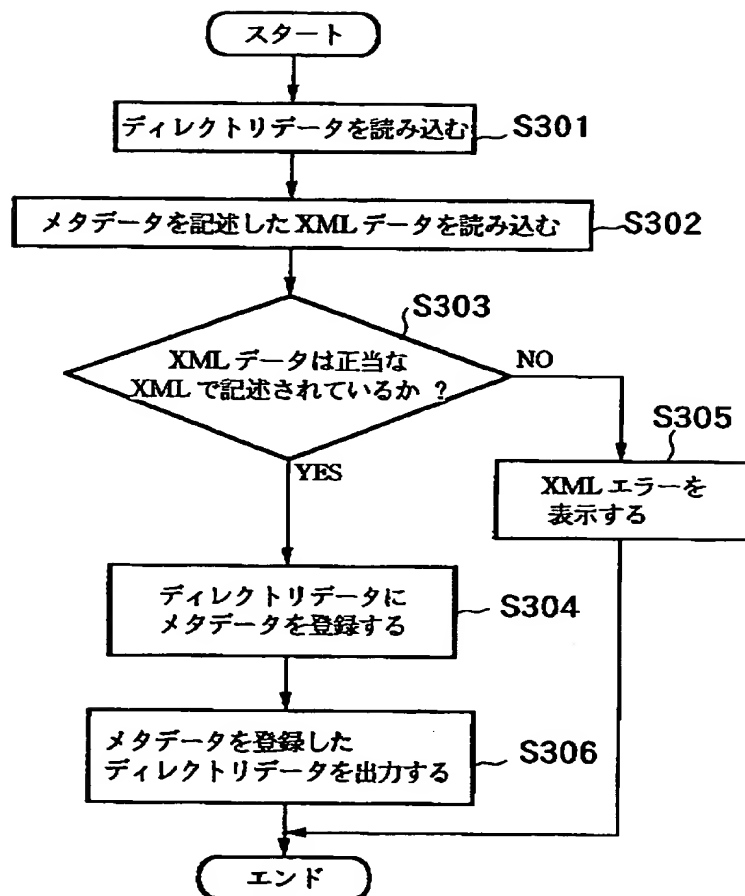


【書類名】 図面

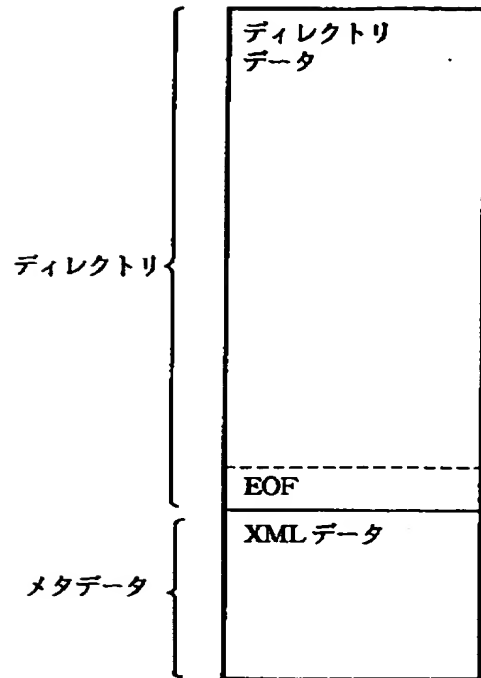
【図 1】



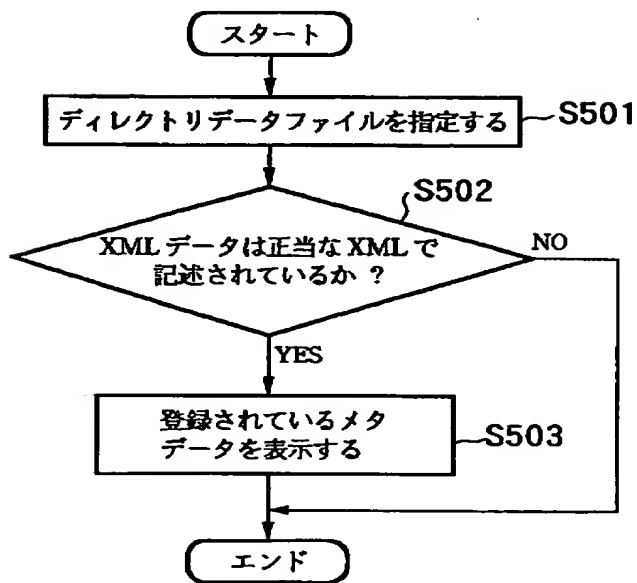
【図 2】



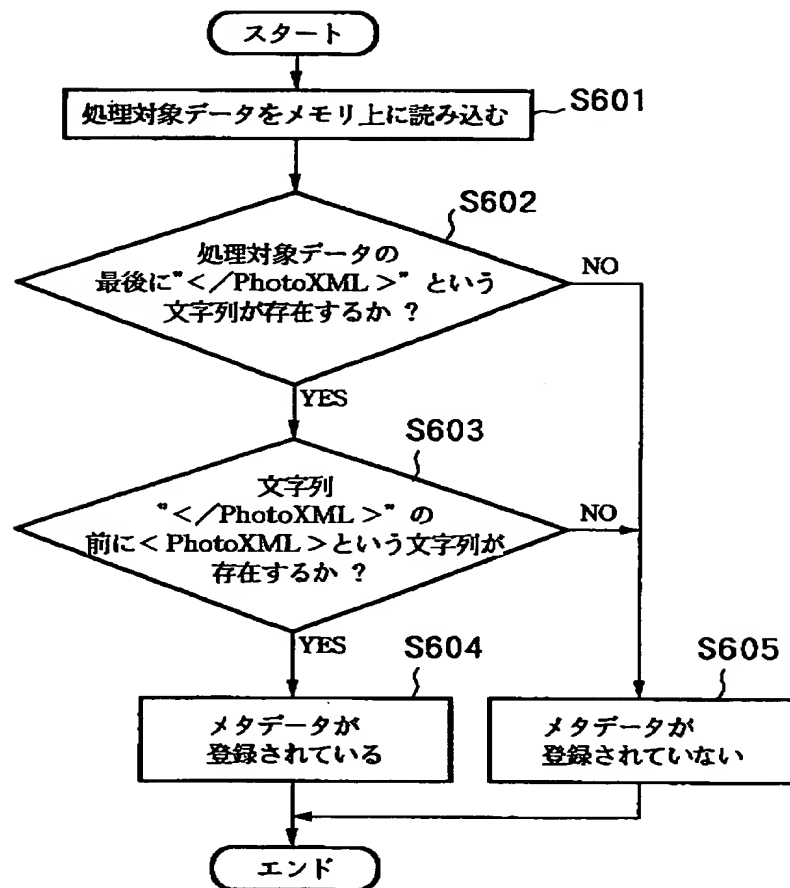
【図 3】



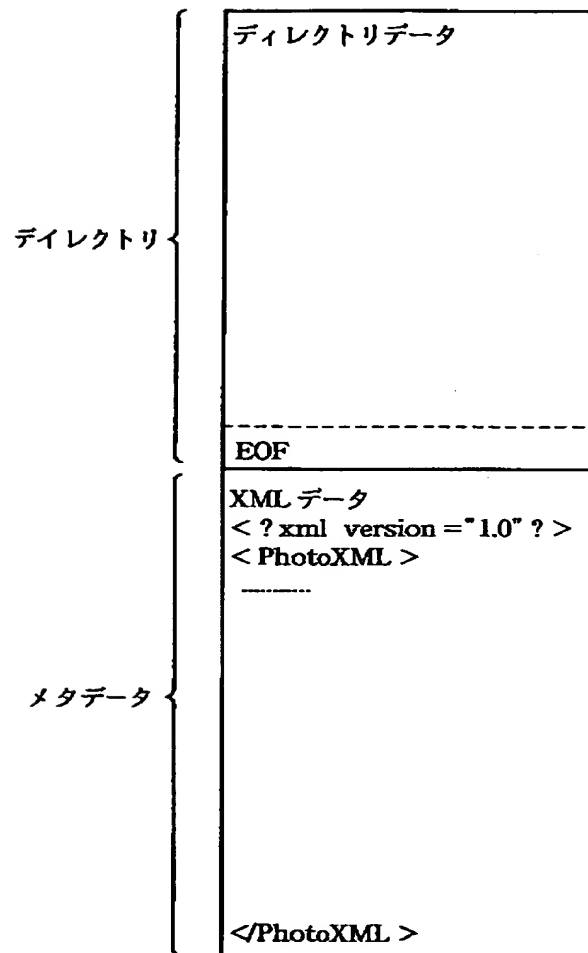
【図 4】



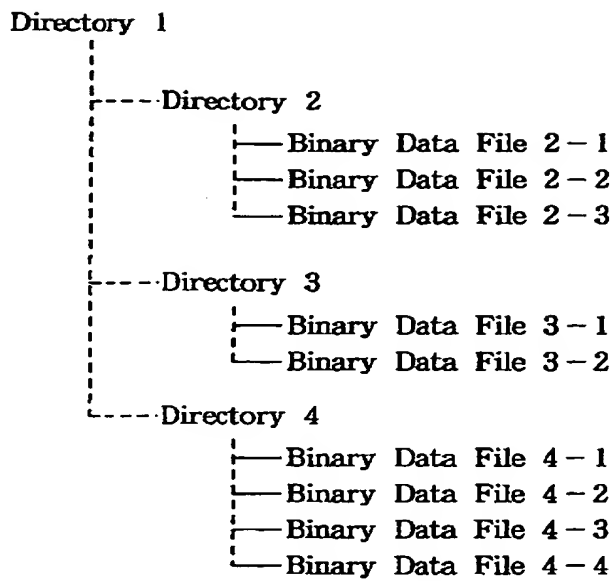
【図 5】



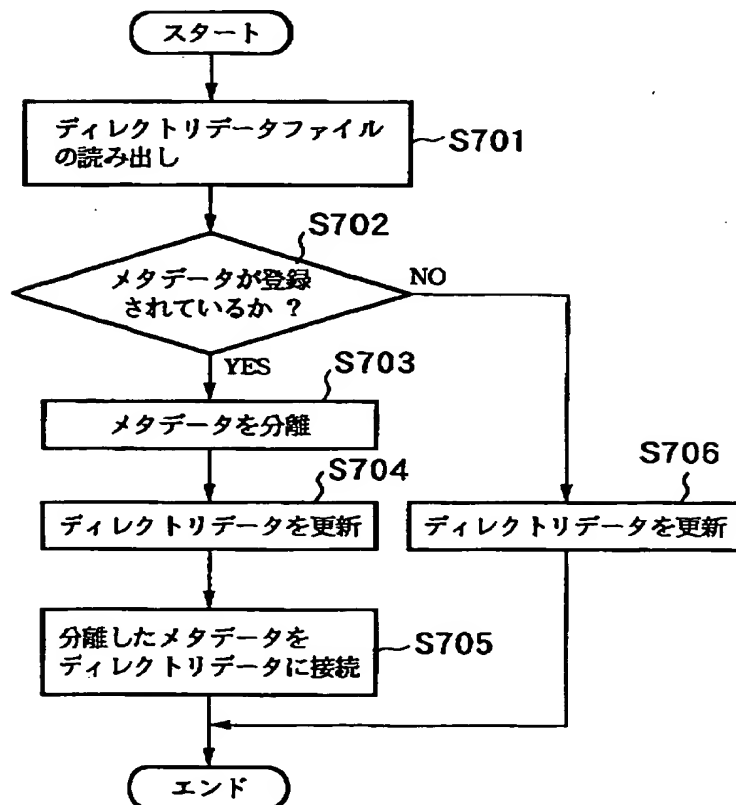
【図 6】



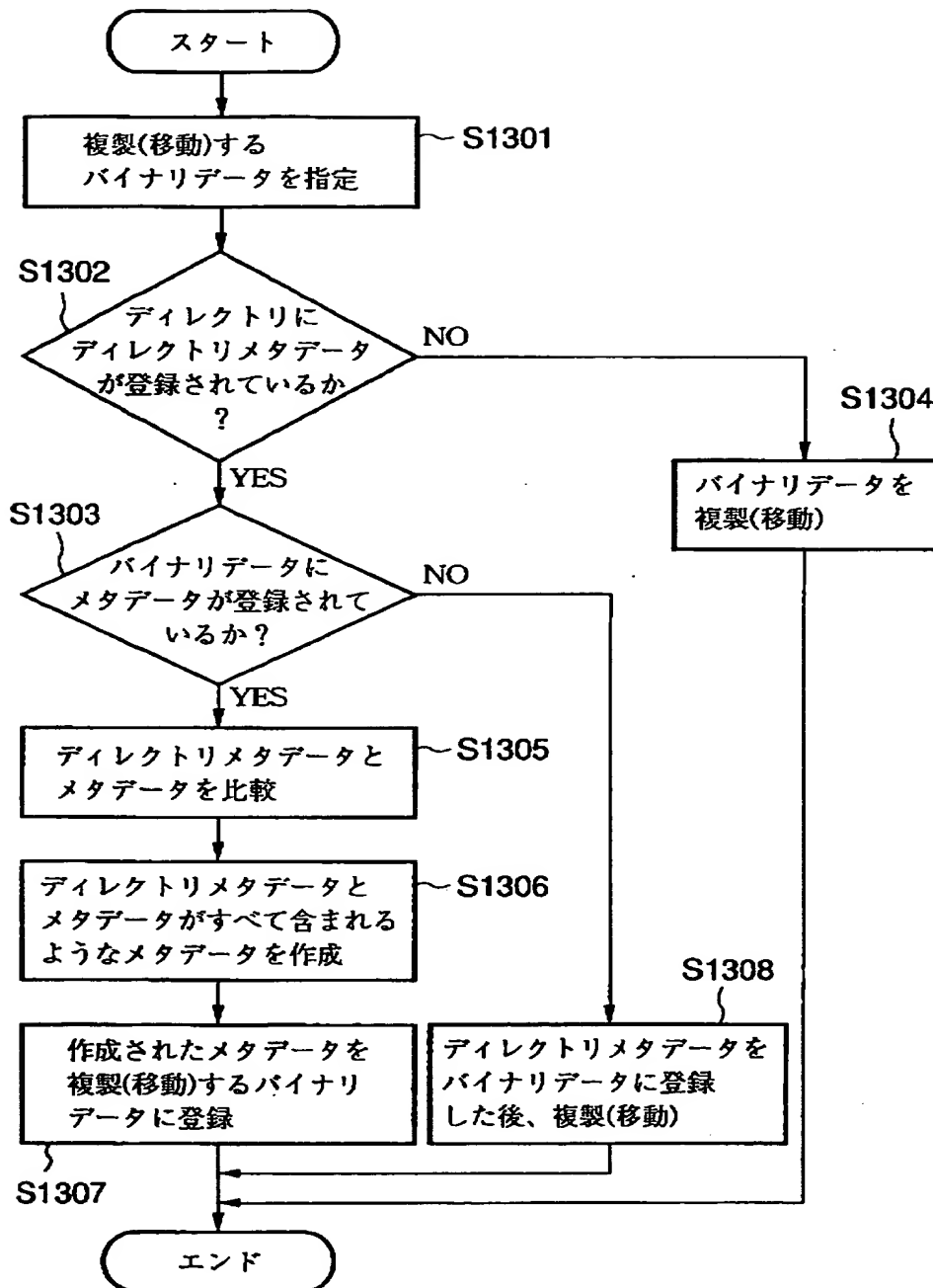
【図 7】



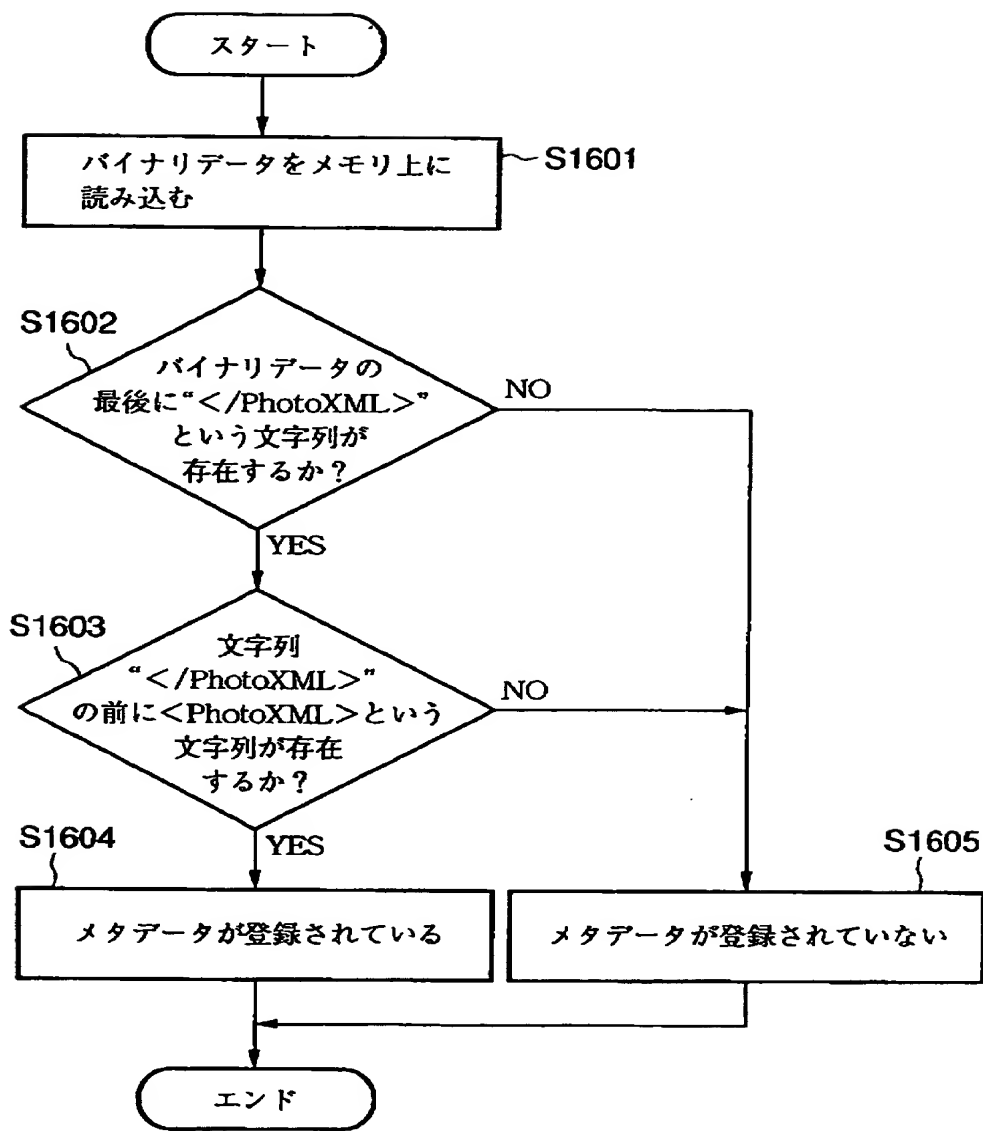
【図 8】



【図 9】

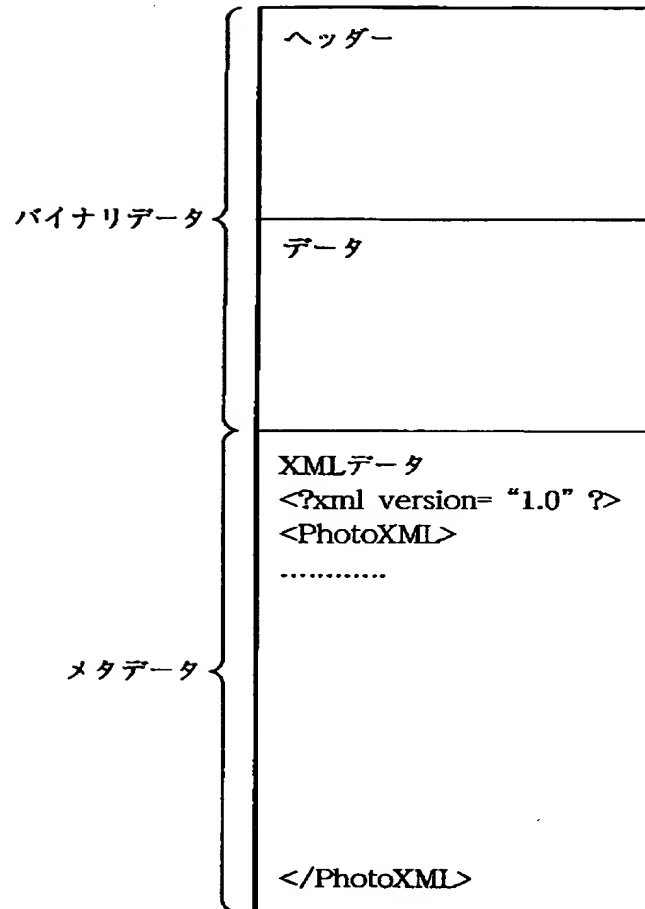


【図 10】

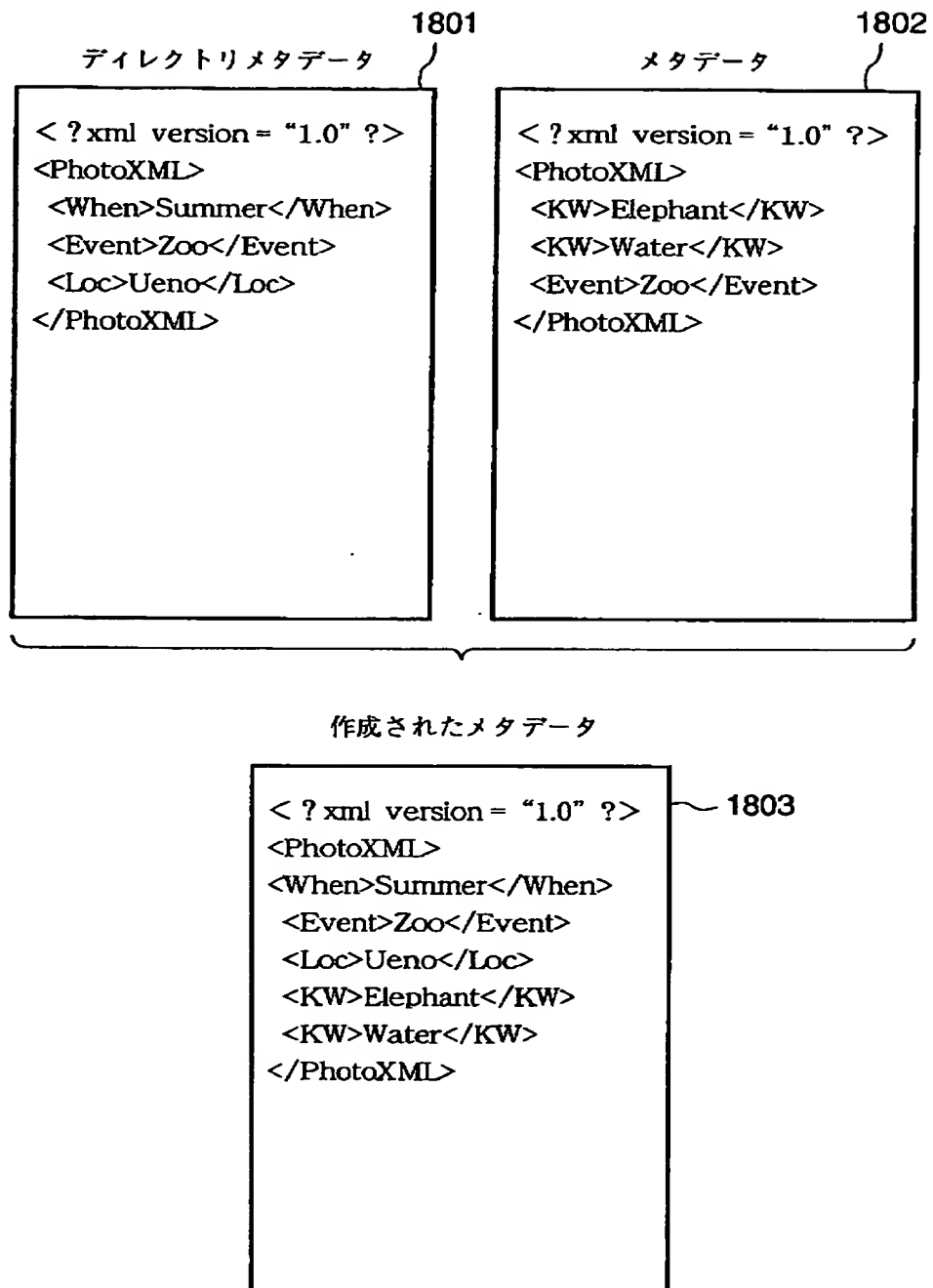




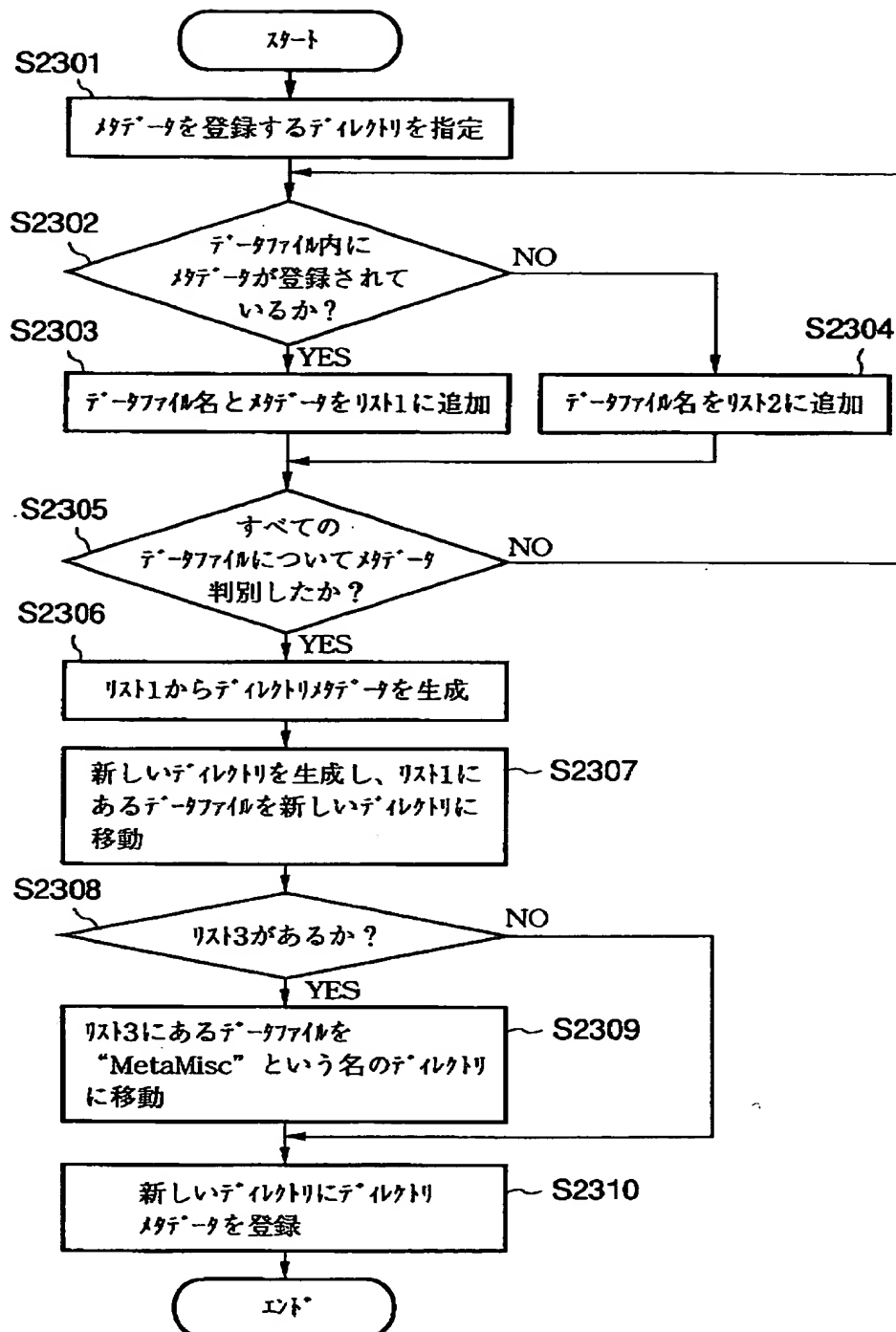
【図 1 1】



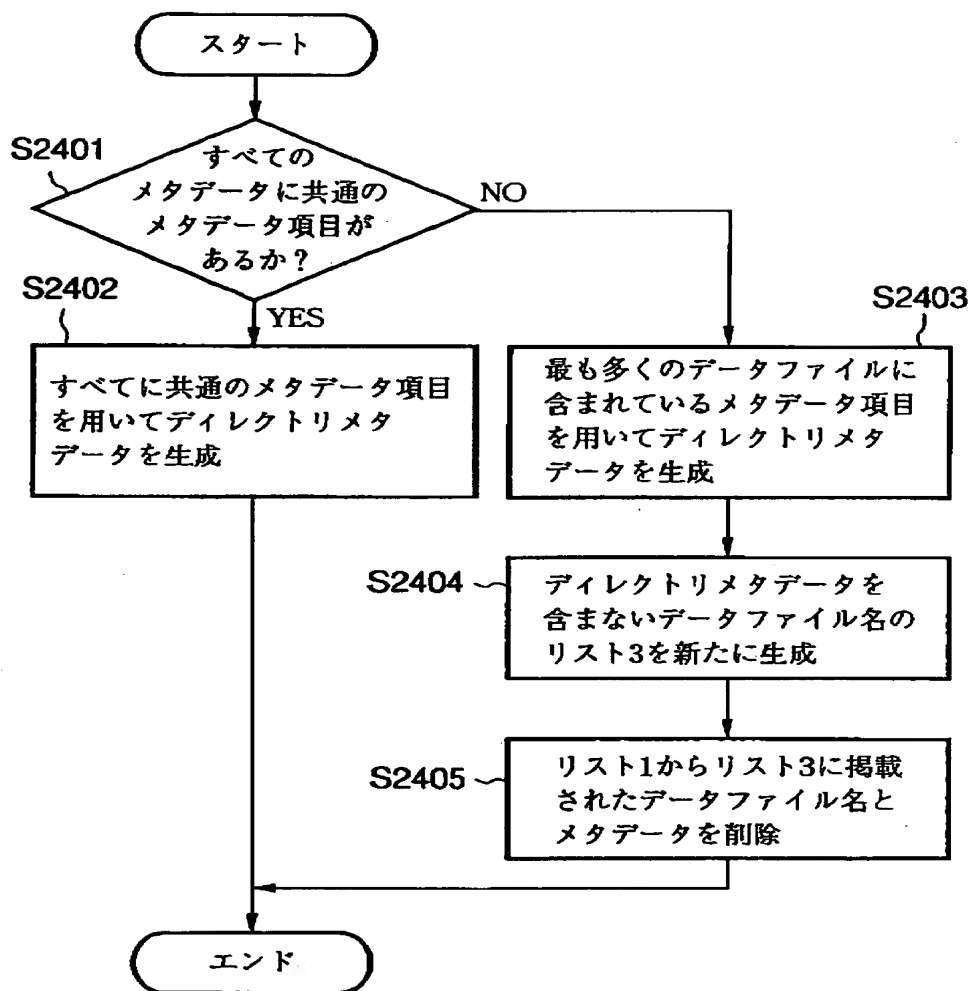
【図 1 2】



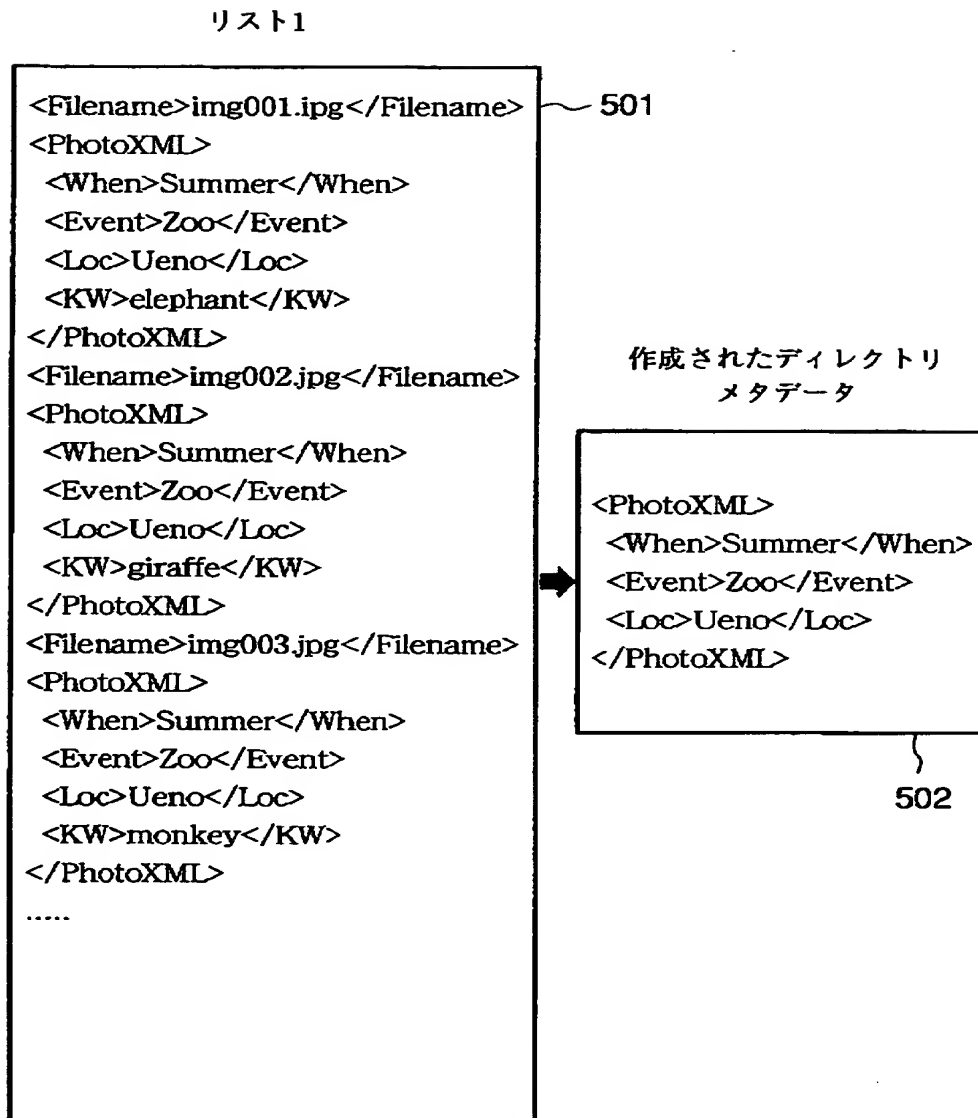
【図13】



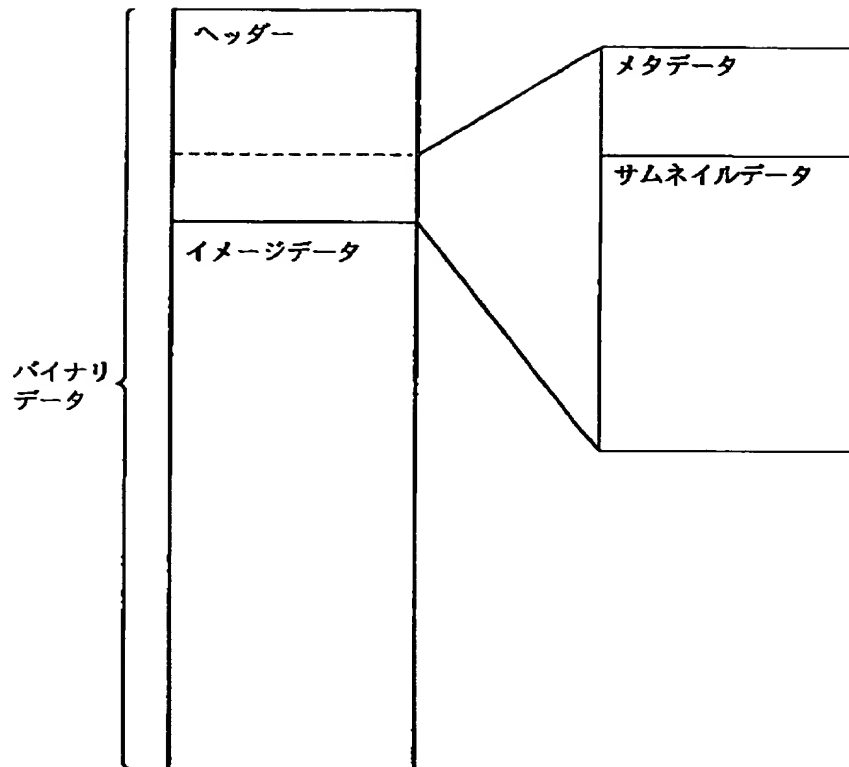
【図 1 4】



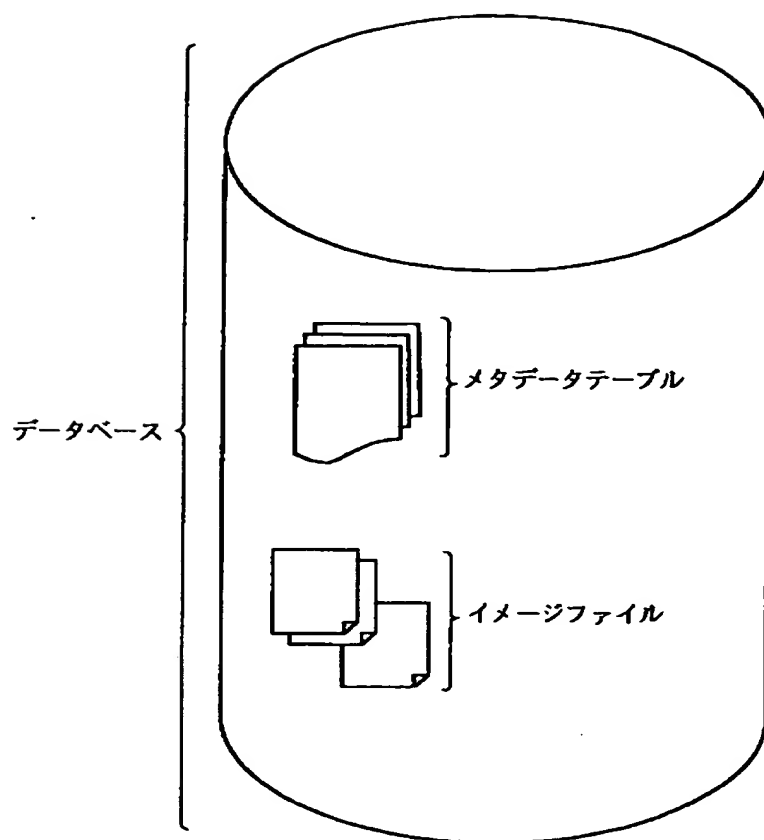
【図 1 5】



【図 1 6】



【図 17】



【書類名】 要約書

【要約】

【課題】 メタデータをディレクトリデータの最後に接続することにより、ディレクトリで管理されている既存のデータに対し、既存のアプリケーションに影響を与えずに、ディレクトリ単位でメタデータを登録可能とする。

【解決手段】 ステップ S 3 0 1 において、ディレクトリ構造でファイルを管理するファイル管理システムによって各ディレクトリ毎に用意されているディレクトリデータを読み込む。ステップ S 3 0 2 で、上記ディレクトリデータに付与すべきメタデータを読み込む。ステップ S 3 0 3 において、上記読み込まれたメタデータが正当な XML で記述されていると判断された場合、ステップ S 3 0 4 へ進み、上記ステップ S 3 0 1 読み込まれたディレクトリデータの後に、ステップ S 3 0 2 で読み込まれたメタデータを接続し、ステップ S 3 0 6 において、得られたデータの全体をディレクトリデータファイルとして出力する。

【選択図】 図 2



認定・付加情報

特許出願の番号	特願 2 0 0 0 - 1 0 9 9 2 3
受付番号	5 0 0 0 0 4 5 8 5 2 6
書類名	特許願
担当官	第七担当上席 0 0 9 6
作成日	平成 1 2 年 4 月 1 7 日

<認定情報・付加情報>

【特許出願人】

【識別番号】	000001007
【住所又は居所】	東京都大田区下丸子 3 丁目 3 0 番 2 号
【氏名又は名称】	キヤノン株式会社

【代理人】

申請人

【識別番号】	100076428
【住所又は居所】	東京都千代田区紀尾井町 3 番 6 号 秀和紀尾井町 パークビル 7 F 大塚国際特許事務所
【氏名又は名称】	大塚 康德

【選任した代理人】

【識別番号】	100101306
【住所又は居所】	東京都千代田区紀尾井町 3 番 6 号 秀和紀尾井町 パークビル 7 F 大塚国際特許事務所
【氏名又は名称】	丸山 幸雄

【選任した代理人】

【識別番号】	100115071
【住所又は居所】	東京都千代田区紀尾井町 3 番 6 号 秀和紀尾井町 パークビル 7 F 大塚国際特許事務所
【氏名又は名称】	大塚 康弘

出 願 人 履 歴 情 報

識別番号 [000001007]

1. 変更年月日 1990年 8月30日

[変更理由] 新規登録

住 所 東京都大田区下丸子3丁目30番2号

氏 名 キヤノン株式会社